RECEIVED
CENTRAL FAX CENTER

OCT 0 4 2005

WOOD, HERRON & EVANS, L.L.P.

BRUCE TITTEL
DONALD F. FREI
DAVID S. STALLARD
J. ROBERT CHAMBERS
GREGORY J. LUNN
KURT L. GROSSMAN
CLEMENT H. LUKEN, JR.
THOMAS J. BURGER
GREGORY F. AHRENS
WAYNE L. JACOBS
KURT A. SUMME
KEVIN G. ROONEY
KEITH R. HAUPT
THEODORE R. REMAKLUS
THOMAS W. HUMPHREY
SCOTT A. STINEBRUNER
DAVID H. BRINKMAN
BEVERLY A. LYMAN, PH.D.
KRISTI L. DAVIDSON

JOSEPH R. JORDAN
C. RICHARD EBY
DAVID E. PRITCHARD
KATHRYN E. SMITH
P. ANDREW BLATT, Ph.D.
DAVID E. JEFFERIES
WILLIAM R. ALLEN, Ph.D.
JOHN PAUL DAVIS
DOUGLAS A. SCHOLER
BRETT A. SCHATZ
DAVID W. DORTON
SARAH OTTE GRABER
STEVEN W. BENINTENDI, Ph.D.
RANDALL S. JACKSON, JR.

OF COUNSEL
JOHN D. POFFENBERGER
DAVID J. JOSEPHIC
THOMAS W. FLYNN
J. DWIGHT POFFENBERGER, JR.
BRADLEY D. BECK

2700 CAREW TOWER
441 VINE STREET
CINCINNATI, OHIO 45202-2917
TELEPHONE: 513-241-2324
FACSIMILE: 513-241-6234
WEBSITE:: www.whepatent.com

PATENT, TRADEMARK, COPYRIGHT
AND UNFAIR COMPETITION LAW
AND RELATED LITIGATION

EDMUND P. WOOD   1923-1968
TRUMAN A. HERRON   1936-1978
EDWARD B. EVANS   1936-1971

October 4, 2005

FACSIMILE COVER SHEET

To: Examiner Chrystine Pham
Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22213-1450

Fax: 571-273-8300
Enclosures:
    Fax Cover Sheet containing Certificate of
    Facsimile Transmission (1 page)
    Appeal Brief Transmittal containing Certificate
    of Facsimile Transmission and Authorization
    to Charge Deposit Account 23-3000 in the
    amount of $500.00 for the Fee (2 pages)
    Appeal Brief (17 pages, including Cover Sheet,
    13 pages Appeal Brief and 3 pages Claims
    Appendix)

From: Scott A. Stinebruner
    Reg. No. 38,323

Re: U.S. Patent Application
    Serial No.   09/997,990
    Filed: November 30, 2001
    Applicant: Jeremy Alan Arnold et al.
    Art Unit: 2192
    Confirmation No.:   4258
    Our Ref: IBM/193

Pages: 20 (including cover sheet)

MESSAGE/COMMENTS
OFFICIAL

CERTIFICATE OF FACSIMILE TRANSMISSION
I hereby certify that this correspondence and the enclosures noted herein (20 total pages, including cover sheet) are being transmitted via facsimile transmission to Examiner Chrystine Pham,, Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 at 571-273-8300 on October 4, 2005.

Judith L. Volk   _October 4, 2005_
Judith L. Volk   Date

RECEIVED
OIPE/IAP
OCT 0 5 2005

ATTY. DOCKET NO. IBM/193

PATENT   Confirmation No. 4258

Judith L. Volk   _October 4, 2005_

    Judith L. Volk   Date

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Jeremy Alan Arnold et al.   Art Unit: 2192

Serial No.:   09/997,990   Examiner: Chrystine Pham

Filed: November 30, 2001   Atty. Docket No.: IBM/193

For: OBJECT-ORIENTED CREATION BREAKPOINTS

Cincinnati, Ohio 45202   October 4, 2005

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF (PATENT APPLICATION-37CFR 191)

1.   Transmitted herewith is the APPEAL BRIEF in this application with respect to the Notice of Appeal received by the Office on August 4, 2005.

2.   **STATUS OF APPLICANT**

      This application is on behalf of

       XX   other than a small entity

       ___   small entity

           Verified Statement:

       ___   attached

       ___   already filed

3.   **FEE FOR FILING APPEAL BRIEF**

      Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is:

       ___   Small entity $250.00

       XX   Other than a small entity $500.00

Page 1 of 2
Serial No. 09/997,990
Transmittal for Appeal Brief of October 4, 2005
IBM Docket No.: ROC920010095US1
WH&E Docket: IBM/193
K:\ber\192\Appeal Brief Transmittal.wpd

4. **EXTENSION OF TIME**

Applicant petitions for an extension of time under 37 C.F.R. 1.136(a) for the total number of months checked below:

Months		Fee for other than small entity	Fee for small entity
_____	one month	$ 120.00	$ 60.00
_____	two months	.... 450.00	.... 225.00
_____	three months	..... 1,020.00	.... 510.00
_____	four months	...... 1,590.00	.... 795.00
_____	five months	...... 2,160.00	...... 1,080.00

Fee: $ _____

If an additional extension of time is required, please consider this a petition therefor.

5. **TOTAL FEE DUE**

The total fee due is:

Appeal brief fee __$500.00__

Extension fee _____

6. **FEE PAYMENT**

_____ Attached is a check in the sum of $ _____

__XX__ Charge fee of $500.00 to Deposit Account No. 23-3000.

7. **FEE DEFICIENCY**

__XX__ Charge any additional extension fee required or credit any overpayment to Deposit Account No. 23-3000.

WOOD, HERRON & EVANS, L.L.P.

By _____

Scott A. Stinebruner
Reg. No. 38,323

2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
(513) 241-2324

Page 2 of 2
Serial No. 09/997,990
Transmittal for Appeal Brief of October 4, 2005
IBM Docket No.: ROC920010095US1
WH&E Docket: IBM/193
K:\ibm\193\Appeal Brief Transmittal.wpd

PATENT

Attorney Docket No. IBM/193
Confirmation No. 4258

UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

_____

Ex parte Jeremy Alan Arnold and John Matthew Santosuosso

_____

Appeal No. _____

Application No. 09/997,990

_____

APPEAL BRIEF

_____

PATENT

RECEIVED
CENTRAL FAX CENTER

OCT 0 4 2005

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant: Jeremy Alan Arnold et al.   Art Unit: 2192
Serial No.: 09/997,990   Examiner: Chrystine Pham
Filed: November 30, 2001   Atty. Docket No.: IBM/193
For: OBJECT-ORIENTED CREATION BREAKPOINTS

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

I. REAL PARTY IN INTEREST

This application is assigned to International Business Machines Corporation, of Armonk,

New York.

II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 14-22, 34-37 and 40 are pending in the Application, and are now on appeal.

Claims 1-13, 23-33 and 38-39 have been canceled.

IV. STATUS OF AMENDMENTS

An Amendment After Final was filed concurrently with the Notice of Appeal on

August 4, 2005, canceling claims 1-6, 8-13, 23-27, 29-33 and 38-39.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The claimed subject matter currently under appeal is generally directed to a computer system, program product and method for debugging an object-oriented computer program by tracking the creation of objects by a plurality of creators (e.g., constructor methods) of a selected class. (Application, p. 5, ll. 22-26).

Debuggers are software tools that can be used to diagnose computer programs and trace errors that arise during execution of the program. One debugging tool commonly supported by a debugger is a breakpoint, which is typically set on a particular statement in a computer program such that, when that statement is encountered during execution, execution of the program is halted to enable a developer to inspect the state of the program at that particular point. In some instances, breakpoints may be specified to be "conditional" so that execution is suspended by a breakpoint only when a particular condition associated with that breakpoint is met (e.g., after the breakpoint has been hit X times). (Application, p. 1, l. 6 to p. 2, l. 8).

One limitation of statement-based breakpoints, however, arises as a result of the unique nature of modern object-oriented programming languages. In an object-oriented program, objects are dynamically created at runtime from templates referred to as "classes", which associate collections of declared data with sets of operations capable of being performed on that data. In a working program, objects are instantiated, or created as needed, and built from the templates defined by their respective classes. During runtime, it is often desirable to provide each new object with initial data and/or initiate particular operations with the object. For this reason, many object-oriented environments support special methods known as constructors, or creators, that are called upon an object's creation. One or more constructor methods are typically defined in each class, while a default constructor method may be defined for some classes when no explicit method is defined by the developer. As a result, objects based upon a particular class may be created by different constructor methods associated with the same class. (Application, p. 2, l. 9 to p. 3, l. 15).

The use of multiple constructor methods for a class provides significant flexibility for programmers. However, the flexibility in object creation has shortcomings during debugging of the computer program. For example, in some situations a programmer may wish to track the

number of objects that are created for a particular class. As an example, a programmer may wish to verify that an excessive number of objects are not created for a particular purpose, e.g., in a database environment, where a programmer might intend for no more than 10 database connections to be active at any given time. Through the use of manually-set breakpoints in all constructor methods, a programmer could manually count the number of object creations; however, doing so could be unduly burdensome when tens, hundreds or thousands of objects are normally created in a program. (Application, p. 3, l. 16 to p. 4, l. 6).

In the alternative, a programmer could utilize conditional breakpoints in each constructor method to trigger only after a certain number of hits. However, given that each breakpoint would independently track the number of times it was hit, and given that a programmer may not know the relative frequency that each constructor method for a particular object is called, the programmer would still not be able to be notified after a specific number of objects were created. (Application, p. 4, ll. 7-13).

The claimed subject matter recited in independent claims 14, 34 and 40 addresses this difficulty by tracking the number of object creations of a class defined in the object-oriented computer program, and halting execution of the object-oriented computer program in response to the number of object creations meeting a particular condition. Of note, the tracked number of object creations includes object creations resulting from multiple creators for the class, such that the condition is based upon the collective object creations resulting from multiple creators for the class. Support for this claimed subject matter is found, for example, in the Application at p. 5, ll. 22-26, and p. 8, l. 30 to p. 9, l. 7.

One manner of implementing such tracking is through the use of a counter that is incremented in response to hitting any of a plurality of breakpoints set on the various creators for the class. (Application, p. 9, ll.1-4). The condition may be based upon a threshold such that execution is halted whenever the value of the counter meets or exceeds the threshold defined for the condition. (Application, p. 15, ll. 7-8).

In addition, in many embodiments, a debugger is configured to automatically identify each creator for a particular class and associate breakpoints with all or a user-specified subset of creators to facilitate tracking, such that any of the breakpoints may trigger a halting of execution

during debugging. (Application, p. 5, ll. 6-12). Furthermore, in some embodiments, the plurality of breakpoints may be associated with a "creation" breakpoint, whereby user input received from a programmer or other user is directed toward performing operations on the creation breakpoint, rather than the breakpoints set on the various creators. Functionality within the debugger thus manages the plurality of breakpoints as a collective group. Irrespective of whether the plurality of breakpoints are associated with a common creation breakpoint, however, through the automated identification of creators for a class in response to user input, a programmer or other user is relieved of the burden of manually identifying creators and setting individual breakpoints on the different creators for a particular class. (Application, p. 5, ll. 13-21).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 14-15, 17-22, 34-35 and 40, stand rejected under 35 U.S.C. § 102(b) as being unpatentable U.S. Patent No. 5,560,009 to Lenkov et al. (hereinafter *Lenkov*) in view of U.S. Patent No. 5,321,828 to Phillips et al. (hereinafter *Phillips*).

B. Claims 16 and 36-37 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over *Lenkov* in view of *Phillips* further in view of U.S. Patent No. 5,754,839 to Pardo et al. (hereinafter *Pardo*).

VII. ARGUMENT

Applicants respectfully submit that the Examiner's rejections of claims 14-22, 34-37 and 40 are not supported on the record, and should be reversed.

A. **Claims 14-15, 17-22, 34-35 and 40 were improperly rejected as being unpatentable over *Lenkov* in view of *Phillips*.**

The Examiner argues that claims 14-15, 17-22, 34-35 and 40 are obvious over *Lenkov* in view of *Phillips*. However, a *prima facie* showing of obviousness requires that the Examiner establish that the differences between a claimed invention and the prior art "are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art." 35 U.S.C. §103(a). Such a showing requires that all

claimed features be disclosed or suggested by the prior art, along with objective evidence of the suggestion, teaching or motivation to combine or modify prior art references, as "[c]ombining prior art references without evidence of such a suggestion, teaching or motivation simply takes the inventor's disclosure as a blueprint for piecing together the prior art to defeat patentability -- the essence of hindsight." In re Dembiczak, 50 USPQ2d 1614, 1617 (Fed. Cir. 1999).

Applicants respectfully submit that neither *Lenkov* nor *Phillips* discloses the various features recited in claims 14-15, 17-22, 34-35 and 40, and as such, the rejections thereof should be reversed. Applicants will hereinafter address the various claims that are the subject of the Examiner's rejection in order.

Independent Claims 14, 34 and 40

Independent claim 14 generally recites a computer-implemented method of debugging an object-oriented computer program. The method includes tracking a number of object creations of a class defined in the object-oriented computer program during debugging, and halting execution of the object-oriented computer program in response to the number of object creations meeting a condition. The claim further recites that the tracked number of object creations includes object creations resulting from multiple creators for the class.

Independent claims 34 and 40 are similar in scope to claim 14, but respectively recite an apparatus and a program product comprising program code that is configured to debug an object-oriented computer program by tracking a number of object creations of a class defined in the object-oriented computer program during debugging, and halting execution of the object-oriented computer program in response to the number of object creations meeting a condition. As with claim 14, each of claims 34 and 40 additionally recites that the tracked number of object creations includes object creations resulting from multiple creators for the class, and a signal bearing medium bearing the program code.

It is important to note, therefore, that the number of object creations that are tracked in each of claims 14, 34 and 40 includes object creations resulting from multiple creators for the class, and furthermore that this tracked number, which incorporates object creations from

multiple creators, is compared to a condition to determine whether execution of a computer program should be halted.

As Applicants discussed in the Amendment and Response dated December 3, 2004, the claimed subject matter provides an advantage over prior breakpoint implementations that could be used to set track object creations by setting individual conditional breakpoints on constructor methods. In particular, with conditions associated with individual breakpoints set on multiple constructor methods, a user would still not be able to halt execution after a specific number of object creations. As an example, suppose a user wanted to halt execution of a program after 10 objects of a particular class were created, where that class had three constructor methods A, B and C. Setting individual breakpoints, even with each having a condition specified therefor to trigger only after a certain number of hits, would not enable a user to be certain that the program would be halted after 10 objects were created, as during runtime, the number of creations resulting from each creator could be vastly different. For example, 10 objects could be created just from constructor method A, B, or C, or any combination thereof (e.g., A=3, B=3, C=4, or A=1, B=9, C=0, or A=0, B=5, C=5, etc.) In essence, it is the sum of the number of hits to the breakpoints set on the different constructors that is of interest to the user, a number that has conventionally never been tracked.

In rejecting claims 14, 34 and 40, the Examiner relies on the combination of *Lenkov* and *Phillips*; however, it is important to note that the Examiner admits, at page 9 of the final Office Action that *Lenkov* does not disclose tracking a number of object creations and halting execution of a program in response to the number of object creations meeting a condition. Indeed, the only disclosure in *Lenkov* that is relevant to breakpoints is found at col. 29, lines 16-44. First, col. 29, lines 16-30 discloses the concept of an "instance breakpoint," which is set on a particular function in a class so that the breakpoint is triggered only if the function is invoked on a particular instance of the class. Second, col. 29, lines 31-36, discloses the concept of a "trigger breakpoint," which is set at an exit point in an object so that the breakpoint will trigger upon execution leaving an object. These breakpoints are apparently used in connection with instance breakpoints to automatically remove an instance breakpoint upon exit from an object.

Third, col. 29, lines 37-44 discloses the concept of a "class breakpoint," which allows a user to set a breakpoint on all member functions of a particular class with a single command. By doing so, execution may be halted whenever any of a class's member functions is invoked.

Presumably, if a class breakpoint is set on a class that has multiple creators or constructor methods, a breakpoint would be set on every creator in a class, along with every other method defined for the class. Nonetheless, the reference is entirely silent with respect to the concept of tying a condition to a class breakpoint, much less a condition that is based upon the total number of hits to different methods in a class. As such, there is no objective evidence within *Lenkov* that would motivate one of ordinary skill in the art to modify the class breakpoints of *Lenkov* to incorporate the ability to track the number of object creations resulting from multiple creators of a given class.

The Examiner apparently relies on *Phillips* for allegedly disclosing tracking a number of hits to a plurality of breakpoints. In connection with the rejection of claim 9 (which the Examiner refers to in the rejection of claim 14), the Examiner apparently relies on the abstract, col. 26, line 40 to col. 27, line 25, and col. 28, line 1 to col. 29, line 25, for allegedly teaching this concept. The abstract, however, mentions only that multiple breakpoints may be used, and does not disclose or suggest the concept of tracking a number of hits to a plurality of breakpoints, as asserted by the Examiner.

Similarly, the passage at col. 26, lines 40-64 mentions only that a breakpoint can have a condition associated therewith. The passage at col. 26, line 67 to col. 27, line 25 generally discloses a number of operations for setting breakpoints, among which is a "break . . . if <cond>;" however, the passage does not discuss any particular condition other than "stop only if the value is nonzero." The passage at col. 28, lines 1-40 discloses the disabling of breakpoints, but is otherwise irrelevant to the concept of a condition associated with a breakpoint. Of note, none of these passages addresses tracking a number of hits to a plurality of breakpoints, as asserted by the Examiner.

The passage at col. 28, line 42 to col. 29, line 25 discloses a number of different conditions that may be applied to a breakpoint. One operation that may be used to set a condition on a breakpoint is a "condition <bnum> <expression>" command; however, there is no

discussion of the condition other than that it can be a boolean expression based upon target system address and/or data. Another operation that may be used to set a condition is an "ignore" command, which specifies a count value such that every time the breakpoint associated with the "ignore" is hit, the count value is decremented. The breakpoint is not triggered, however, until the count value reaches a zero value.

While *Phillips* discloses the concept of setting a condition on a breakpoint that triggers the breakpoint only after N hits, this disclosure falls short of disclosing or suggesting the concept of tracking the number of object creations for a class that result from multiple creators, as required by each of claims 14, 34 and 40. Indeed, *Phillips* discloses little more than what Applicants have already admitted was conventional at p. 4, ll. 7-13 of the Application.

Claims 14, 34 and 40 require that execution of a program be halted when the tracked number of object creations, which includes object creations resulting from multiple creators, meets a condition. The condition is therefore compared against a number that represents object creations resulting from multiple creators.

Phillips, on the other hand, discloses only tracking the number of hits to an individual breakpoint, which is set at a single location in a program, and thus cannot be associated with multiple creators of a class. As such, there is no objective evidence within *Phillips* that would motivate one of ordinary skill in the art to modify the class breakpoints of *Lenkov* to incorporate the ability to track the number of object creations resulting from multiple creators of a given class.

The Examiner has cited no other evidence of motivation in the art to modify *Lenkov* to incorporate any such functionality, and as such, the Examiner has failed to meet the burden required to establish a *prima facie* case of obviousness. Indeed, the combined teachings of *Lenkov* and *Phillips* simply fail to disclose or suggest that a number representative of object creations from multiple creators for a class be tracked, or that a condition for halting execution of a program can be tested against such a tracked number. At most, the combination of *Lenkov* and *Phillips* suggests the addition of an ignore condition (as taught by *Phillips*) to each of the individual breakpoints set on the member functions of a class (including all of the creator functions) as a result of a user setting a class breakpoint (as taught by *Lenkov*). As discussed

above, however, providing separate conditions on individual breakpoints associated with different creators does not enable a user to track the total number of object creations, or to halt execution after a desired number of object creations, as is supported by the subject matter recited in claims 14, 34 and 40.

Accordingly, Applicants submit that the Examiner has failed to establish a *prima facie* case of obviousness as to any of claims 14, 34 and 40, and that the rejections thereof should be reversed. Furthermore, as none of the other art of record discloses or suggests such claimed subject matter, Applicants respectfully request allowance of claims 14, 34 and 40, as well as of claims 15-22 and 35-37 which depend therefrom.

Dependent Claim 17

Claim 17 depends from claim 14, and additionally recites the concept of, in response to user input, identifying a plurality of creators for a class and setting a plurality of breakpoints on the identified creators. As such, claim 17 discloses the automated setting of multiple breakpoints to implement the tracking of object creations from multiple creators of a class.

In rejecting claim 17, the Examiner refers to the rejections of claims 1-4 and 12, which rely principally upon *Lenkov* for allegedly disclosing the concept of identifying creators for a class. The relevant portions of *Lenkov* (specifically, the discussion of a class breakpoint at col. 29, lines 37-44), however, disclose setting breakpoints on all member functions of a class. Of note, however, by setting breakpoints on all member functions, rather than just on the constructor methods or creators for a class, *Lenkov* cannot be interpreted as disclosing "identifying a plurality of creators for [a] class," as required by claim 17. *Lenkov*, if anything, arguably identifies all of the member functions of a class, but does not discriminate between creator and non-creator functions.

There is no suggestion in the reference of the desirability of identifying the creators of a class, as opposed to all member functions. Indeed, given that the purpose of a class breakpoint, as disclosed in *Lenkov*, is to halt execution whenever an object of a particular class is encountered, regardless of how, modifying *Lenkov* to identify only the creators for a class would completely alter the operation of the class breakpoint.

It is possible that the Examiner may be taking the position that by identifying all member functions of a class, *Lenkov* is inherently identifying all creators for the class. Applicants respectfully submit, however, that "identifying" (i.e., "establishing the identify of," or "ascertaining the origin, nature, or definitive characteristics of")[1] one type of entity that is generic to different species does not specifically "identify" either species. As such, just because *Lenkov* identifies all member functions, *Lenkov* does not specifically identify member functions that are creators, just as it does not identify member functions that are not creators.

Accordingly, Applicants submit that one of ordinary skill in the art would not interpret *Lenkov* as disclosing or suggesting "identifying" the creators for a class, as required by claim 17. Accordingly, the Examiner has not established a *prima facie* case of obviousness as to claim 17, the rejection thereof should be reversed.

Dependent Claim 19

Claim 19 depends from claim 17, and additionally recites the concept of, after identifying the plurality of creators, displaying a list of the identified creators and receiving user input to select a subset of identified creators, such that the plurality of breakpoints are set on only the subset of the identified creators.

In rejecting claim 19, the Examiner refers to the rejections of claims 1-4 and 12, among which the rejection of claim 3 is the most relevant. This rejection relies on col. 29, lines 55-67 of *Lenkov* for allegedly disclosing displaying a list of identified creators after identifying the creators for a class, and receiving user input to select a subset of the identified creators.

This passage in *Lenkov*, however, only discloses that overloaded functions can be displayed, and that a user can set a breakpoint on all overloaded functions. *Lenkov* does not disclose the concept of identifying all creators for a class, as discussed above in connection with claim 17. Furthermore, even if *Lenkov* did disclose the identification of creators, there is no disclosure in the reference of receiving user input to select a subset of identified creators such that breakpoints are only set on the subset of identified creators. The cited passage specifically

---

[1] Definitions obtained from The American Heritage® Dictionary of the English Language, Fourth Edition. Houghton Mifflin Company, 2004.

recites that a user can set a breakpoint on "all" overloaded functions, so this passage cannot be relied upon for disclosing or suggesting providing a user with a list of functions and allowing the user to select of subset of the functions, irrespective of whether those functions are creators.

Accordingly, Applicants submit that one of ordinary skill in the art would not interpret *Lenkov* as disclosing or suggesting receiving user input to select a subset of identified creators for a class such that breakpoints are only set of the subset of creators, as required by claim 19. Accordingly, the Examiner has not established a *prima facie* case of obviousness as to claim 19, the rejection thereof should be reversed.

Dependent Claims 15, 18, 20-22 and 35

Claims 15, 18, 20-22 and 35 are not argued separately.

B.   Claims 16 and 36-37 were improperly rejected as being unpatentable over *Lenkov* in view of *Phillips* and further in view of *Pardo*.

Applicants respectfully submit that none of *Lenkov*, *Phillips* and *Pardo* discloses the various features recited in claims 16 and 36-37, and as such, the rejections thereof should be reversed. Applicants will hereinafter address the various claims that are the subject of the Examiner's rejection in order.

Dependent Claim 16

Claim 16 depends from claim 14, and additionally recites that tracking the number of object creations includes incrementing a counter in response to hitting any of a plurality of breakpoints set on a plurality of creators for the class.

In rejecting claim 16, the Examiner admits that neither *Lenkov* nor *Phillips* discloses the concept of incrementing a counter in response to hitting any of a plurality of breakpoints. Instead, the Examiner relies on Pardo, and in particular Fig. 2, col. 5, lines 37-50 and col. 6, lines 15-60. Pardo, however, merely discloses the use of a counter to track watchpoints for the purpose of detecting when a processor state should be stored in a history buffer. Pardo deals in particular with the fact that a watchpoint may be hit during execution of a program on a pipelined

processor, but that due to speculative execution, the instruction that the watchpoint is associated with may be flushed or canceled before execution (see col. 1, lines 54-60). The counter is thus used to determine when a watchpoint detected by a pipelined processor should essentially be ignored.

Other than the fact that Pardo discloses a counter that is incremented whenever different watchpoints are detected, there is absolutely no suggestion in the reference of the desirability of using a counter to track multiple breakpoints associated with a particular class, much less to track multiple breakpoints associated with multiple creators. Furthermore, there is no suggestion in the reference that a condition could be tested against the value of the counter. No objective evidence has been presented that would motivate one of ordinary skill in the art to modify *Lenkov* to incorporate a counter to track the hitting of breakpoints associated with multiple creators for a class.

Absent any evidence of motivation, the rejection amounts to little more than hindsight-based analysis. Accordingly, Applicants respectfully submit that the Examiner has not established a *prima facie* case of obviousness as to claim 16, and that the rejection thereof should be reversed.

Dependent Claims 36 and 37

Claim 36 depends from claim 34, and additionally recites that the program code is configured to track the number of object creations by incrementing a counter in response to hitting any of a plurality of breakpoints set on a plurality of creators for the class, and that the program code is further configured to, in response to user input, identify the plurality of creators for the class and set the plurality of breakpoints on the identified creators.

As discussed above in connection with claim 16, none of the cited references discloses or suggests the use of a counter to track the hitting of breakpoints set on multiple creators for a class. Furthermore, as discussed above in connection with claim 17, none of the cited references discloses or suggests the concept of "identifying" a plurality of creators for a class in response to user input, and setting breakpoints on the identified creators. Accordingly, Applicants respectfully submit that claims 36, and claim 37 which depends therefrom, are non-obvious over

the cited references for the same reasons presented above for claims 16 and 17. Reversal of the Examiner's rejections of claim 36 and 37 are therefore respectfully requested.

VIII. CONCLUSION

In conclusion, Applicants respectfully request that the Board reverse the Examiner's rejections of claims 14-22, 34-37 and 40, and that the Application be passed to issue. If there are any questions regarding the foregoing, please contact the undersigned at 513/241-2324. Moreover, if any other charges or credits are necessary to complete this communication, please apply them to Deposit Account 23-3000.

Respectfully submitted,

WOOD, HERRON & EVANS, L.L.P.

Date: **4 OCT 2005**

By: _____
Scott A. Stinebruner
Reg. No. 38,323

2700 Carew Tower
441 Vine Street
Cincinnati, Ohio 45202
(513) 241-2324

Claims Appendix: Claims on Appeal 09/997,990

IX. CLAIMS APPENDIX: CLAIMS ON APPEAL (S/N 09/997,990)

1. - 13. (Canceled)

14. (Once Amended) A computer-implemented method of debugging an object-oriented computer program, the method comprising:

    (a) tracking a number of object creations of a class defined in the object-oriented computer program during debugging, wherein the tracked number of object creations includes object creations resulting from multiple creators for the class; and

    (b) halting execution of the object-oriented computer program in response to the number of object creations meeting a condition.

15. (Original) The method of claim 14, wherein the condition is the number of object creations meeting or exceeding a threshold.

16. (Original) The method of claim 14, wherein tracking the number of object creations includes incrementing a counter in response to hitting any of a plurality of breakpoints set on a plurality of creators for the class.

17. (Original) The method of claim 14, further comprising, in response to user input, identifying the plurality of creators for the class and setting the plurality of breakpoints on the identified creators.

18. (Original) The method of claim 17, wherein identifying the plurality of creators includes identifying every creator for the class.

19. (Original) The method of claim 17, further comprising, after identifying the plurality of creators, displaying a list of the identified creators and receiving user input to select a subset of identified creators, wherein the plurality of breakpoints are set on only the subset of the identified creators.

- A-1-

20. (Original) The method of claim 17, wherein the plurality of breakpoints are collectively set on all of the identified creators in response to the user input.

21. (Original) The method of claim 17, wherein identifying the plurality of creators and setting the plurality of breakpoints are performed in response to user input to set a creation breakpoint, and wherein the plurality of breakpoints are associated with the creation breakpoint.

22. (Original) The method of claim 18, wherein each creator comprises a constructor method defined in the class.

23. - 33. (Canceled)

34. (Once Amended) An apparatus, comprising:

(a) a memory within which resides at least a portion of an object- oriented computer program; and

(b) program code configured to debug the object-oriented computer program by tracking a number of object creations of a class defined in the object-oriented computer program during debugging, and halting execution of the object-oriented computer program in response to the number of object creations meeting a condition, wherein the tracked number of object creations includes object creations resulting from multiple creators for the class.

35. (Original) The apparatus of claim 34, wherein the condition is the number of object creations meeting or exceeding a threshold.

36. (Original) The apparatus of claim 34, wherein the program code is configured to track the number of object creations by incrementing a counter in response to hitting any of a plurality of breakpoints set on a plurality of creators for the class, and wherein the program code is further configured to, in response to user input, identify the plurality of creators for the class and set the plurality of breakpoints on the identified creators.

- A-2-

Claims Appendix: Claims on Appeal 09/997,990

37. (Original) The apparatus of claim 36, wherein the program code is configured to identify the plurality of creators and set the plurality of breakpoints in response to user input to set a creation breakpoint, and wherein the plurality of breakpoints are associated with the creation breakpoint.

38. - 39. (Canceled)

40. (Once Amended) A program product, comprising:

 (a) program code configured to debug an object-oriented computer program by tracking a number of object creations of a class defined in the object-oriented computer program during debugging, and halting execution of the object-oriented computer program in response to the number of object creations meeting a condition, wherein the tracked number of object creations includes object creations resulting from multiple creators for the class; and

 (b) a signal bearing medium bearing the program code.

- A-3-